The Motor Industry Software Reliability Association

MISRA c/o Electrical Group, MIRA, Watling Street, Nuneaton, Warwickshire, CV10 0TU, UK.
Telephone: (024) 7635 5290. Fax: (024) 7635 5070. E-mail: misra@mira.co.uk Internet: http://www.misra.org.uk

Report 7

Subcontracting of Automotive Software

February 1995

PDF version 1.0, January 2001

Acknowledgements

The following were contributors to this report:

Keith Longmore, Lotus Engineering
David Newman, Ford Motor Company Ltd
John Fox, The Centre for Software Engineering Ltd
Neil Andrewartha, Rolls-Royce and Associates Ltd
David Ward, MIRA (final compilation and editing)

Summary

This report is not intended to be a treatise on the legalities of writing contracts. It is intended to be an engineer's view of the topics which should be considered by engineers, managers, project managers and purchasing departments involved with software products—buying, selling, creating, managing. It merely sets out to explain **why** these topics should be considered.

The report is divided into three major parts:

- software aspects of subcontracting in the automotive industry
- technical aspects of contracts
- commercial aspects of contracts.

Recommendations

- The following should be defined in the invitation to tender:
  - Establish a glossary of terms used, with their definitions as they are applied in the contract
  - What the product of the project is expected to do, and what it is expected to not do
  - Specify, using definitive references, any legislative constraints (e.g FMVSS standards) on the product
  - What parameters are to be visible to diagnostics, and what priority structure is to be assigned to identified errors and warnings
  - Accessibility of diagnostic information: protocols, authorities, tools, in-vehicle display, connector standard
  - Failure management policy—definition of criteria for initiating default states; default state definitions; management of user warnings; legislative constraints on default states (permissible/not permissible)
  - Reliability—definition and criteria, targets
  - Development lifecycle—what model (e.g. 'v'), standard (e.g. IEC SC65A), milestone dates
  - Quality plan—accreditation, timing, documentation, assessment requirements
  - Procedures for agreeing the above—including dispute and query rectification.

- Adoption of an agreed product development and quality lifecycle plan is beneficial to both parties to the contract—it should not be treated as a punitive device.

- It is in both parties' interest to agree the accuracy of the requirements specification at the point in the lifecycle which offers least risk to them both.

- In order for the supplier to understand his responsibility for quality and safety, risk assessment procedures should be defined in the contract.

- Both parties must be fully cognisant of the required tools for the product, and agree a mechanism to ensure that if accredited tools are required, **only** those tools are used.

- It is essential where proprietary software is specified to be used that commercial, timing and any technical constraints are understood, agreed, and documented by both parties, at both the Invitation to Tender and contract stages.

- If it is proposed by either party to reuse software, or use library software, this should be addressed in the contract, with agreed provisions for validation issues.

- Route of communication, and authorities, as well as any specific technical requirements for project staff should be identified, where appropriate, by name, in the contract.

- It is in the interest of both parties to define problem reporting and corrective action procedures in the contract.

- Progress reporting criteria, as well as procedures, should be laid down in the contract, so that problems are identified at the earliest point.

- It is very important that acceptance criteria, based on joint supplier/purchaser project/design reviews, are contractually defined in an unambiguous manner.

- Maintenance and support must not be treated as a simple warranty matter: they should be planned for, and be contractually defined, including the level and type of staff, period of support, and man-hours required.

- Be clear who owns what when material is free-issued to a contractor, or purchased by the contractor for the purpose of carrying out the project.

- It is in the Purchaser's best interest to ensure that proprietary software and protocol terms are complied with by all contractual parties, and that, should a contractor be prosecuted for licence infringement, that a protection or contingency mechanism may be extremely important for the Purchaser.

- Terms and conditions for transfer of licence agreements must be fully understood by all parties concerned, and if purchaser's software is involved, the terms should be covered in the contract.

- Advice should be drawn from the references regarding patents in respect of software, and/or from a patent lawyer or legal consultant. It is important to be aware that actual code and algorithms cannot be patented, only processes which the code is carrying out. It is also vital to be aware that a significantly greater risk regarding patent infringement exists elsewhere, especially in the USA.

- Advice on copyright and IPR should be drawn from the references or from a legally qualified person.

- If knowhow is sensitive, ensure that a contractor cannot obtain a complete view of the project by modularization of the project, and use multiple contractors, also by ensuring that the contract imposes a procedure for controlling the passing of all information in writing to the contractors.

- It may be desirable for some critical or sensitive software to be subject to a copy being lodged with a suitable escrow agent. This may be relevant if there is a risk of a contractor's failure or some legal action after the software has been put into service.

- Fitness for purpose is most likely to be solely the responsibility of the Purchaser; this should be a convincing argument for proper specification and quality procedures to be imposed—and practised by the Purchaser, as well as the contractor(s).

- If proper procedures are imposed by contract, and there is insurance that they are followed, there should be little question of a negligence claim becoming a reality.

- Performance bonds are counterproductive in the software contracting arena, and generally cannot be recommended practice. If proper procedures are contractually agreed, there should be no need of such punitive measures.

- Penalty clauses should be largely irrelevant if proper procedures are called for and adhered to. They may be counterproductive in resolving any problems which arise, especially, late in a contract.

- Post delivery support should be treated on the same basis as design/development procedures, and be defined in the contract. It should not be unlimited.

- Liabilities should be defined as far as possible in the contract, especially where proprietary software is concerned. However, the ultimate liability must be considered to lie with the Purchaser for ensuring that the product is fit for the marketplace, and for ensuring that contracts require adherence to proper practices.

- It is up to the Purchaser to ensure that both he and the contractor take all the steps recommended by MISRA, and compliance with software standards such as ISO 9000, as appropriate, if the risk of falling foul of consumer protection legislation is to be minimized.

Contents

Page

Part I — Software Aspects of Subcontracting in the Automotive Industry

1. Summary

1.1 Summary of key subcontract concerns

- Software is still relatively new to automotive engineering.
- It is often perceived as merely part of a "black box", rather than a specific entity requiring any special treatment. Perception of software development costs is often poor.
- Inspection and quality procedures are usually process oriented, so software is perceived as part of a process, which is intangible and invisible.
- A large part of software normally provides rarely used hardware failure management and diagnostics facilities.
- Special contractual considerations need to be given to aspects such as confidentiality, liability, security, etc. This is especially relevant to the supply of data to non-franchised and roadside repair organizations.

1.2 Summary of examples of subcontract situations

- Direct Purchaser–Supplier relationship
- Multiple sourcing
- Consultancy and independent audit
- Software tools and proprietary software
- Retrospective use of software.

All of the above points are significant issues in relation to subcontracting of software.

2. Examples of subcontract situations

2.1 Direct purchase of vehicle systems

This is the simplest form of subcontract, where, typically, a vehicle manufacturer places a contract with a supplier for supply of a system complete with software. In this situation there may be specific software costs involved for the purchaser, and the main diversification of contract type relates to the way in which these costs are paid by the purchaser:

- costs amortized over a specified number of production units
- costs "built in" to the unit cost of development units
- part of a fixed price development contract.

There may be contractual conditions specific to the cost recovery scheme which relate payment for software to performance or acceptance procedures.

2.2 Multiple sourcing

It is often the case in the vehicle industry that dual or multiple sourcing is applied to some components or systems. This may be because of a desire to promote competitive pricing, or because of fears regarding continuity of supply.

It may become contractually very complex where multiple supply is involved, especially if software is further subcontracted. There are four potential situations:

- vehicle manufacturer separately contracts software and hardware
- multiple suppliers for hardware use the same software supplier
- multiple suppliers of both hardware and software
- multiple serial supply chains, where each prime contractor subcontracts various aspects of the work, including, possibly, multiple software sources for software modules.

See MISRA Report *Diagnostics and Integrated Vehicle Systems* [12].

2.3 Consultancy and independent audit

There may be contractual requirements placed on the supplier by the vehicle company to employ a nominated consultant or independent software or system auditor. Conversely, the supplier may wish to employ such services on his own behalf as part of the normal development process. This clearly raises issues such as confidentiality and IPR, for which special contract provisions may be required. It is also possible that the principle of escrow may be called for in order to protect the purchaser in the event the vendor goes into receivership or defaults in some way.

2.4 Software tools and proprietary software

There are specific issues relating to licensing and liability/warranty concerning the use, purchase, and fitness for purpose of software tools, which it is essential to understand, particularly in the light of consumer protection legislation. Where a vendor has purchased a software tool specifically for a purchaser's project, the purchaser may require the tool to be handed over on completion of the project, or require that a license be granted to allow the purchaser to be able to use the same tool.

The purchaser may require that tools be validated by a third party, which could if carried out, be in breach of the terms of the software proprietor's license agreement.

See "Modelling and CASE tools" in [14].

2.5 Retrospective use of software

Often suppliers will have core software in their products which may have been designed speculatively, or for another client. This raises two issues:

- amortization of costs
- acceptability of standards of validation applied when the software was written—which may be some years ago.

3. Software and the vehicle industry—what makes it special?

3.1 A relatively new science in the vehicle industry

Products employing embedded software have been in use in the vehicle industry for a little over ten years. Compared with most other disciplines employed in the vehicle industry this is quite immature. However, as the cost of computing has fallen dramatically in recent years, the scope of usage in vehicles has grown swiftly. The understanding of the issues involved in the use of software based products may not have kept pace with the rate of growth of their usage in some areas.

3.2 Perception

The vehicle industry is firmly entrenched in buying "black boxes" and externally designed and made components. Usually, mechanical or electrical components can be inspected at the point of entry to the vehicle assembly plant; software, however, cannot: it is invisible and intangible. Because of the mechanical/electrical bias, electronic systems tend to be regarded as components—"black boxes"—bought against a functional specification, and only tested against that functional specification. There is therefore, particularly in buying departments, little perception of software as a specific entity which may need separate validation and specialized treatment. It is often not perceived as a significant cost in the development or production of a unit, even though it may indeed be a major development cost.

Because the industry’s approach to quality is process oriented, so, often, is its approach to testing of software based products.

3.3 Failure management

A great deal of the software effort and content in typical vehicle systems is concerned with the primary safety of the vehicle in the event of a malfunction of the electronics; therefore much of the embedded software is concerned with rarely used diagnostic and failure

management routines. This inevitably means that functional testing will not invoke many of the failure states which are potentially possible.

3.4 Confidentiality and servicing

Confidentiality is an important factor in the highly competitive arena of the vehicle industry. It is complicated by the need for repair either at the roadside, or by non-franchised workshops (brought about by free competition laws). In particular software confidentiality can be difficult to maintain, and special care is needed at the contract stage in handling this issue. It also has significance if independent audit is a specification requirement.

3.5 Liability

Liability may not be a clear-cut issue, and special care needs to be exercised to understand who is liable for what, and what limitations are to be agreed. This is a significant issue in respect of any tools employed, which may be licensed on the basis of a disclaim of any liability for their correctness or consequences of their malfunction.

Part II — Technical Aspects of Subcontracts

4. Technical aspects of subcontracts

4.1 Summary

A contract to develop a product involving software should not be seen as a punitive document aimed at extracting penalties when the project goes wrong. Rather, it should seek to ensure that the project does **not** go wrong.

This can only be achieved by good communication between all parties involved, and by addressing technical issues as either part of, or an annex to, the commercial contract aspects, and ensuring that all parties to the contract have a clear definition of technical, quality, and problem reporting/corrective action procedures. Such a document must ensure that suitable contingency is encapsulated in the technical provisions of the document.

Total Quality Management must be treated as complementary to such a contract, and its procedural requirements, not a replacement for it.

4.2 The development lifecycle

4.2.1 Rationale

Frequently, engineering projects involving software fail to meet their objectives because of poor communications between supplier and purchaser: expectations do not match the supplier's interpretation of what is required. **Interpretation** is the key word—a functional specification must **not** require, or be open to, interpretation, by either party to a contract. This is of special importance—many contracts carry a clause which mandates that the specification is the ultimate authoritative document for a project, above even the contract itself, if there happens to be conflict between the two documents. This is especially the case where standard contracts are used.

The Purchaser should create a formal Invitation to Tender incorporating a functional specification, agreed with the Engineering Departments involved, and Purchasing Department, for issue to a Supplier. This should not be a verbal contact, nor a minuted meeting alone (although minutes are a legal document), since errors and omissions may go unnoticed. What should be defined in the invitation to tender?

- Establish a glossary of terms used, with their definitions as they are applied in the contract
- What the product of the project is expected to do, and what it is expected to not do
- Specify, using definitive references, any legislative constraints (e.g FMVSS standards) on the product

- What parameters are to be visible to diagnostics, and what priority structure is to be assigned to identified errors and warnings

- Accessibility of diagnostic information: protocols, authorities, tools, in-vehicle display, connector standard

- Failure management policy—definition of criteria for initiating default states; default state definitions; management of user warnings; legislative constraints on default states (permissible/not permissible)

- Reliability—definition and criteria, targets

- Development lifecycle—what model (e.g. 'v'), standard (e.g. IEC SC65A), milestone dates

- Quality plan—accreditation, timing, documentation, assessment requirements

- Procedures for agreeing the above—including dispute and query rectification.

Development lifecycle and quality plan are further discussed below.

4.2.2 Development and quality lifecycle

Contracts usually contain a timing requirement, even if only a required finish date for the work contracted. Software development is generally acknowledged to be very difficult to accurately estimate, and frequently is underestimated. This may be disastrous in a project where the contract has no milestones, and quite catastrophic if a penalty clause is included for late delivery. It must be stressed that enforcing a deadline can be very damaging to the health of a software product, and may also result in considerably more corrective work than would have been necessary earlier in the development process. It is advisable to incorporate some flexibility and contingency into the lifecycle plan.

- Adoption of an agreed product development and quality lifecycle plan is beneficial to both parties to the contract—it should not be treated as a punitive device.

The most important item in the development lifecycle is the output from requirements analysis, and the specification of any requirements analysis tools (e.g. CORE). This should ensure that the Functional Requirements are converted to a formal (**not** necessarily "Formal" as in Formal Methods) specification for software (and hardware as appropriate). The contract should require that the formal software specification is created and validated, in a form which is accessible to audit: this should be the first milestone in the contract. By requiring this milestone, with audit by an appointed software auditor, areas of the specification which are ambiguous, incorrect, or may give rise to problems later in the project may be identified and addressed at a point in the life of the contract which offers least risk to both parties.

- It is in both parties' interest to agree the accuracy of the requirements specification at the point in the lifecycle which offers least risk to them both.

An agreed Hazard Analysis should normally be a requirement defined in the contract, based on Controllability and Integrity levels (see MISRA Report *Integrity* [13]). This is important in ensuring that the supplier understands his responsibility for the system's safety and quality procedures. FMEA or FMECA, FTA or other process which is related to HAZAN/HAZOP (Hazard Analysis/Hazards and Operability study) should also be specified.

- In order for the supplier to understand his responsibility for quality and safety, risk assessment procedures should be defined in the contract.

The purchaser will often have preferences for software tools and techniques to be used in the execution of the project. Tools and techniques should be specified in the contract, in an unambiguous manner (Product XX, Version 2.0f, with add-on facility Y, etc.) This is of special importance if the product is to be validated by a third party, or by means of specific tools. The contract may require accredited products to be used, and should this be so, should contain a mechanism to ensure that the accreditation is adhered to, and that any non-accredited copies of the tools are excluded from use in the project without prior approval of the purchaser.

- Both parties must be fully cognisant of the required tools for the product, and agree a mechanism to ensure that if accredited tools are required, **only** those tools are used.

The use of specific software tools will inevitably mean the use of proprietary software. These may be commercial products, or involve the supplier's or purchaser's proprietary software. In the case of commercial products, ownership of the licence must be made clear, along with any arrangements for disposal at the end of the project. (See Commercial Aspects). Other proprietary software may require special contractual arrangements, and also involve training. This should be agreed and defined fully by both the Invitation to Tender, and the subsequent contract. The use of proprietary software may cause technical implications to be identified. These should either be agreed and documented in the contract, or provisions made in the contract for resolving any such issues in a controlled and reasonable manner.

- It is essential where proprietary software is specified to be used that commercial, timing and any technical constraints are understood, agreed, and documented by both parties, at both the Invitation to Tender and contract stages.

Many automotive systems are based on the use of a "library" of software modules (as well as, often, a library of hardware modules). This software may have been developed in the past, to lesser or different quality requirements than the current contract demands. This raises the issue:

- if this software has functioned without reported faults for a significant period,

what would a validation achieve?

- what is the level of confidence in validation not forcing modifications to be needed?
- would modifications lower confidence in the software in other respects?

The contract should contain a contingency for dealing with the results of these questions. The contingency must be designed to address this issue as early as possible in the lifecycle.

- If it is proposed by either party to reuse software, or use library software, this should be addressed in the contract, with agreed provisions for validation issues.

It has already been stated that software projects often fail to meet their objectives because of inadequate communications between the contracting parties. It is strongly suggested that individual staff members in both parties are nominated in the contract for specific duties. This must include **one** point of contact for communications between supplier and purchaser, and authorities on each side, for example signing off each stage such as validation. Any special technical requirements for staff working on the project should also be identified.

- Route of communication, and authorities, as well as any specific technical requirements for project staff should be identified, where appropriate, by name, in the contract.

Problems occur on any engineering project; what is essential is effective reporting and resolution of those that occur. All too often, problems go unresolved because of human factors (loss of face, fear, etc.), lack of procedures ("don't bring me problems, just go away and solve them"), or are solved in an unreported, uncontrolled way (without authority, avoiding revalidation, with unrecorded modifications. All that this approach does is to create the potential for future problems: "bugs"; software not matching documentation; incorrect interface to other software modules. It is vital to establish problem reporting, recording, and corrective action procedures as a matter of contract. These procedures must involve a review process to assess their impact and scope, and to devolve control of problem solving activities. It is usual for this to be partly catered for in a contract by specifying version and modification control procedures, for example, by a password approach to access the software mechanisms needed to change signed-off modules.

- It is in the interest of both parties to define problem reporting and corrective action procedures in the contract.

For a project to run successfully to a predefined timetable, progress reporting is an important activity, especially in software projects. Progress **criteria** should be defined by contract, as well as reporting procedures. There is no point in reporting "I have written 40 lines of code this week"—useless if it does not work correctly. A better criterion would be "I have validated 10 lines of code this week, with a confidence level for correctness of X". It is very easy for difficulties to grow into large problems, because progress criteria and procedures did not reflect the real state of progress.

- Progress reporting criteria, as well as procedures, should be laid down in the contract, so that problems are identified at the earliest point.

Acceptance criteria, which are usually based on project/design reviews between purchaser and supplier, need to be defined clearly, and unambiguously. Next to the functional requirements, the most likely area of misunderstanding and contention is on what basis the product is accepted by the purchaser as compliant with its specification. It is usual for acceptance criteria to be laid down for each milestone defined in the lifecycle plan; there may be, for large projects, an acceptance plan which runs in parallel with the development lifecycle. At each milestone, the project may be subject to audit of work carried out; as well as ensuring that quality is to plan, the audit output may be used as a sign-off authority for, for example, stage payment.

It is usual for projects to have a "Taking into Service" procedure, which usually involves a complete design review; inspection of all audit outputs to ensure that the system has been tested, etc., to the specified requirements; and some form of laid down field tests to establish that the system performs to the purchaser's satisfaction. This procedure is usually the final development lifecycle stage to authorize payment. There still may be a part payment withheld pending satisfactory performance in the field over a specified time period (see Commercial Aspects).

It is very important for both parties that these criteria and procedures to be a contractually defined requirement.

- It is very important that acceptance criteria, based on joint supplier/purchaser project/design reviews, are contractually defined in an unambiguous manner.

Software products usually require maintenance and support after taking into service. Quite often, a purchaser, rather than ensuring a contractual arrangement, "expects" support on an ad hoc basis. A safety critical software project **must** employ version and modification control procedures; these are made nonsense of if maintenance is then an unplanned activity, possibly demanded as a "consumer right" rather than in a contractually defined manner. Warranty may be a condition of contract (see Commercial Aspects) but this should be confined to matters of negligence rather than support and maintenance. Support and maintenance implies changes arising out of experience in the field, not "fixing" warranty-type problems.

It is best for the contract to specify a period of support, and level (technical, authority) of support staff. It is often specified as "X man-hours of senior Engineer, Y man-hours of ...etc" on a call-off basis over the specified period. These staff members should either have worked on the project during the development phase, or else have been specifically trained for the tasks to be performed, by members of the development team.

- Maintenance and support must not be treated as a simple warranty matter: they should be planned for, and be contractually defined, including the level and type of staff, period of support, and man-hours required.

4.2.3 Summary of contract technical recommendations

The above recommendations are intended to aid the process of agreeing contracts for the execution of software projects in the automotive industry. They are compiled from an engineer's, common sense, viewpoint. It may be most appropriate that these issues are enclosed in appendices to the project contract, or perhaps as a schedule to the contract. It is important that technical issues such as these are not mixed up with commercial contractual detail in the contract. However, although they are principally of **technical** rather than commercial importance, most will have commercial implications.

Engineers should be aware of the commercial implications of their actions and requirements, but it is important that there is a balance, especially where software is involved. All too often contractual deadlines are imposed and enforced without consideration of the technical implications of so doing: the result is frequently problems with the software when it is put into service. It is far better for both parties to the contract to design its terms with the possibility and implications of software problems in mind, rather than bloody-minded imposition of contractual conditions which will ultimately have to be modified for the project to succeed. It must be remembered that problems which have to be addressed at a late stage in the project may cost one or two orders of money more than they would have cost if addressed early on; the same is true of the time involved. A useful quote is "It is strange how deadlines and budgets become flexible when the outcome of the project doesn't work in the field".

4.2.4 Total Quality Management

There is a growing use of variants of a culture which can be broadly grouped under the name of "Total Quality Management" (TQM).

The principles embodied in TQM are:

- individual trust, personal responsibility
- minimum bureaucracy
- "everyone is a customer"—so keep them happy
- "right first time".

In general terms, there may be a conflict between the type of contractual issues described in this document and the principles of TQM. In principle, if a purchaser trusts his suppliers to do a good job, there should be no need for a contract; contracts are for resolving problems arising from things going wrong—and if TQM is effective, the product will be "right first time". In practice TQM means ensuring customer satisfaction through product quality—but not necessarily documented proof: trust means not needing to prove you have done it.

It is not that simple where software is concerned.

- The functionality of a mechanical system will be decided by a mechanical engineer, who must then communicate to a software engineer—or team of

software engineers—his requirements. Given the complexity of automotive systems, how can there be assurance that the mechanical engineer's complex requirements are understood, fully, and unambiguously by the software engineer, unless the information is written down in detail, including communications between the parties concerned regarding queries and answers?

- It is widely accepted that software must be written in a structured, disciplined, well documented fashion, if errors are to be minimized. The rigour with which procedures are applied is expected to increase with greater criticality of the functions carried out in software.

- It is widely accepted, and called for in international software standards, that there is evidence of quality by virtue of rigorous documentation. A company could be vulnerable to a claim of negligence in the event of a product liability suit without this evidence.

- Software standards call for traceability in all software creation activities—usually traceability back to the requirements specification.

- In view of the complexity of software, it is not a simple matter to "fix" it if it does not work late in the lifecycle—it can be very costly indeed to do so.

It is important that quality is treated as a culture; it cannot be enforced, added on, or somehow fixed later. Companies which require to adopt a quality assurance regime are well advised to indoctrinate their staff to want to create a quality product. TQM is one way of implementing this; however, it must be stated that the vast majority of companies fail to successfully implement such a regime. All too often, TQM is used as an excuse to avoid documentation, to avoid the kind of procedure called for in software standards: it must not be treated this way. TQM should be an aid to ensuring that quality procedures and documentation are adhered to, not avoided. It should be complementary to procedures such as ISO 9000-3 (TickIT).

There is another aspect of TQM which potentially conflicts with the software quality procedures called for: sign-off responsibility. Generally speaking, companies with a hierarchical structure nominate a chief engineer for some levels of sign-off; director for others; maybe a quality engineer for yet others. TQM companies often adopt a non-hierarchical structure, based on the theory of individual responsibility. This may create a problem if, for example, a non-TQM company subcontracts software to a TQM company with a non-hierarchical structure, since the requisite levels of structured authority will not exist in the TQM company to comply with a contract of the form described.

There is also an inherent risk in the culture of "treat everyone as a customer": in attempting to please the "customer", there will inevitably be at least a temptation, at worst, pressure, to "cut corners" in creating software. Given the cost in terms of time and effort in adhering to a software quality plan, and typical project managers' priorities, this risk is a very real one.

4.2.5 In conclusion—TQM

Whilst the benefits of implementing a TQM-style culture are recognized, it also behoves those responsible for this approach to accept that software does not fit quite so readily into such a culture as more tangible, less complex products. The issues must be handled in such a way as to ensure that the type of approach called for in software quality standards, including MISRA recommendations, is not undermined by the TQM culture, but, rather, is enhanced by it.

Whilst most contracts are created merely to define procedures and penalties for the eventuality of failure, software contracts have a strong technical *raison d'être*. They have a strong role in preventing failure as well. TQM, if applied too literally in this respect, could actually encourage failure of software projects.

Part III — Commercial Aspects of Subcontracts

5. Commercial aspects of subcontracts

5.1 Summary

Software projects carry an often significant commercial risk. To seek to impose an inequitable responsibility for carrying that risk is likely to be at least unhelpful. Most contracts are designed to cater for the eventuality of something going wrong; it is far better to design software contracts to minimize the risk of going wrong.

The contract should cover, fully and unambiguously, all aspects of the software contract which may become contentious, or may place the project at risk. This report lists relevant issues, and highlights those which may be problematic.

In general, this report views subcontracting principally in the context of UK contracts.

5.2 Definitions

Purchaser The party placing a contract; "Purchasing" is the term used in the contract for identifying its proprietor.

Vendor The party on whom the contract is placed; this is the term used in the contract.

Contractor Referred to as the Vendor in the contract. This report uses the term "contractor" since it is considered to be a clearer term to use for the purposes of the report.

Note: Topic references following headings refer to Appendix D.

5.3 Ownership

5.3.1 Supplied and delivered materials [Topic reference C6]

It should be made clear whether materials supplied to a subcontractor become the property of the subcontractor or not. This is particularly important if it involves third party software, where licensing becomes an issue; it may also be important if the contract involves drawings, specifications, third party documentation such as standards, company standards, and EPROMs or other chips—especially if the latter contain software material. It is also important to stipulate the ownership of materials bought by the contractor for the purpose of carrying out the project.

- Be clear who owns what when material is free-issued to a contractor, or purchased by the contractor for the purpose of carrying out the project.

5.3.2 Protocols and proprietary software licences [Topic references C7, A3]

Generally speaking, proprietary software and protocol standards are licensed to their users. The terms of the licence involved **must** be fully understood and complied with by **both** parties to a software contract, especially if the contractual Purchaser issues a piece of proprietary software or protocol to his contractor. In the latter case, the Purchaser should supervise the installation of the proprietary software onto the contractor's computer(s), ensuring that the licence is complied with, and retain ownership of the original discs on which the proprietary software was purchased. If, however, the contractor purchases his own proprietary software/protocols, adherence to the terms of the licence are his responsibility. The Purchaser, in this event, should ensure that the contract he has placed does not accept any responsibility for the contractor's non-compliance with licence terms. It should be borne in mind that, should a contractor break licence terms, and be prosecuted as a result, the penalties can be very severe, resulting in financial embarrassment to the contractor (or, in the extreme, failure of his business), and, perhaps more importantly, delaying the project.

- It is in the Purchaser's best interest to ensure that proprietary software and protocol terms are complied with by all contractual parties, and that, should a contractor be prosecuted for licence infringement, that a protection or contingency mechanism may be extremely important for the Purchaser.

5.3.3 Licence transfer [Topic reference C8]

There are two aspects to this: commercial software, and other proprietary software. In the case of commercial software, the licence agreement is usually straightforward: the licence may be transferred, providing the original discs and manual are passed over with the licence when the software is sold, and copies destroyed (although a few licences still outlaw this). Where a contractor is licensed to use a piece of software produced by the Purchaser, a reason may appear which requires that this licence is transferred to a third party. The contract should specify the conditions under which this is permitted, which signatories are needed for approval, and under what terms.

- Terms and conditions for transfer of licence agreements must be fully understood by all parties concerned, and if purchaser's software is involved, the terms should be covered in the contract.

5.3.4 Patents [Topic reference C4]

Patents in the arena of software present a very unclear situation to software developers, and should be viewed with caution.

It is not possible **in the UK** to patent software, only the application of software (i.e. not the code or algorithm, but the process carried out by the software code is, however, copyrightable). The situation elsewhere, notably in the USA, is very different, and it is very easy to infringe one of the many obscure software patents which exist.

It is strongly recommended to take all reasonable care to assess the likelihood of patent infringement; it may be insured against, but, if the high cost is acceptable, some patent law consultants will carry out appropriate searches, and effectively warrant that patent non-infringement.

If patentable output should issue from any part of a software contract, ownership of the patent must not be in doubt. Ownership should have been already defined in the contract. However, if the contractor infringes someone else's patent, it is quite likely that the Purchaser will ultimately at very least share responsibility. It is important to include a disclaimer in the contract for responsibility for the contractor's wilful, knowing infringement of patent; it is very unlikely that a software contractor will accept any responsibility for unknowing infringement. As with some other issues, a punitive approach is likely to be counterproductive: it is pointless insisting that a contractor assumes responsibility, if the outcome is that his business is ultimately destroyed, with all the consequences for the project inherent in that.

It is also important to be aware that knowingly infringing a patent in the USA can result in a tripled penalty in the courts.

It should be defined in the contract who has responsibility for negotiating with a patent holder, if there is a possibility of using a third party's patented process or design.

- Advice should be drawn from the references regarding patents in respect of software, and/or from a patent lawyer or legal consultant. It is important to be aware that actual code and algorithms cannot be patented, only processes which the code is carrying out. It is also vital to be aware that a significantly greater risk regarding patent infringement exists elsewhere, especially in the USA.

5.3.5 Copyright and IPR [Topic references C3, C2]

Ownership of copyright and intellectual property rights (IPR) is little different from any other contract, and advice is available as to form of contract in the references. It should be noted that code is subject to copyright.

- Advice on copyright and IPR should be drawn from the references or from a legally qualified person.

5.3.6 Knowhow [Topic reference C5]

Treatment of knowhow is little different from any other type of contract; however, for a software engineer to do his job may require that knowhow about the system to be controlled by the software may have to be imparted to the contractor. It is probably wisest to provide this information in **written** form, so that it may be subject to the terms of a confidentiality agreement (see below); however, the contractor may have acquired knowhow as a result of the contract which could result in his setting up to compete with the Purchaser. Provided no

confidential documents or information are involved, or infringement of copyright or patents, the Purchaser cannot easily protect himself against this. It is therefore essential that the purchaser is aware of the risks of passing on knowhow, and lays down rules for transmission of information to the software contractor by his staff. Compilation of a detailed and unambiguous specification, and modularizing the project, so that sections may be contracted individually to a number of contractors, is probably the best defence against allowing knowhow to fall into the hands of a contractor.

- If knowhow is sensitive, ensure that a contractor cannot obtain a complete view of the project by modularization of the project, and use multiple contractors, also by ensuring that the contract imposes a procedure for controlling the passing of all information in writing to the contractors.

5.3.7 Escrow [Topic reference C10]

Sometimes, if there is a risk of a contractor going out of business, or a particular piece of software is sensitive (or, perhaps, a risk of "chipping"—see [14]), a Purchaser may require that a copy of the software is lodged with a third party for safe keeping. This is known as escrow. An example, in the case of software, the third party is the National Computing Centre, who may in the event of a legal action involving software in their keeping be called to offer evidence in a court of law.

- It may be desirable for some critical or sensitive software to be subject to a copy being lodged with a suitable escrow agent. This may be relevant if there is a risk of a contractor's failure or some legal action after the software has been put into service.

5.4 Warranties

5.4.1 Fitness for purpose [Topic reference C19]

In general, consumer law requires that products in the marketplace are fit for their intended purpose. Therefore, if a product involving software is the subject of a consumer action claiming it is not fit for its intended purpose, it will be necessary if the software was subcontracted, to establish where blame should lie—was it software or hardware, contractor or Purchaser? This may be difficult to establish in practice, unless the contractor failed to meet his contractual obligations (e.g. the software does not perform to specification). However, if the contractor has met his obligations, and the problem is, say, due to the interaction of software and hardware, it could be difficult for the Purchaser to claim from the contractor. This is especially so if more than one contractor is involved, and if say, two contractors have been employed, one for hardware, one for software. It should be assumed that fitness for purpose is solely the responsibility of the Purchaser; this should be a persuasive argument for the Purchaser to ensure that the project is properly specified and monitored, verified and validated.

- Fitness for purpose is most likely to be solely the responsibility of the Purchaser; this should be a convincing argument for proper specification and quality procedures to be imposed—and practised by the Purchaser, as well as the contractor(s).

5.4.2 Negligence [Topic reference C17]

Negligence may be difficult to prove: after all, **no** software can be assumed 100% fault free. Manufacturers and vendors have to be able to show that they have exercised their "duty of care". For contractual purposes, quality and organizational procedures (see MISRA Guidelines) should be called for in the contract to ensure that a claim of negligence is able to be properly and confidently defended. (See Liabilities)

- If proper procedures are imposed by contract, and there is insurance that they are followed, there should be little question of a negligence claim becoming a reality.

5.4.3 Performance bonds [Topic reference C20]

In some areas of engineering contracts, it has been a practice for the Purchaser to demand a Performance Bond, often as a condition of the Invitation to Tender. A Performance Bond in is often imposed in addition to a Penalty Clause—see next section.

There are two purposes for a Performance Bond:

- to ensure that only companies with a high level of confidence in their ability, and of sufficient substance to accept the level of risk in the project, will tender

- to ensure that, should a project fail to meet its objectives, the Purchaser has a significant redress.

There is some recognition that the level of commercial risk in software projects renders Performance Bonds counter productive. Performance Bonds do not seem to be normal practice in the vehicle industry, though they have been imposed in some transport sectors.

- Performance bonds are counterproductive in the software contracting arena, and generally cannot be recommended practice. If proper procedures are contractually agreed, there should be no need of such punitive measures.

5.4.4 Penalty clauses and proportional withholding of payments [Topic reference C21]

Penalty clauses are not enforceable in English law. (However, they still appear in contracts in some quarters). They are enforceable elsewhere in the world. In England Liquidated Damages clauses are enforceable; these are designed to limit damages to the extent of actual loss incurred by a breach of the contract, and do not have the purpose of "encouraging" either

party to perform. If a Liquidated Damages clause is written so that the latter interpretation might be placed on it, it, too, will have no force in English law.

Penalty clauses come in largely the same category as performance bonds. A punitive approach may actually cause corrective action to be much more difficult to achieve in the event of a problem occurring, and indeed may only succeed in pushing up costs to the Purchaser: most contractors faced with penalty clauses simply reflect the risk in their tender. They may also lead to a contractor "covering up" known defects where these may cause a project overrun. Proper procedures and quality standards render such approaches largely irrelevant.

- Penalty clauses should be largely irrelevant if proper procedures are called for and adhered to. They may be counterproductive in resolving any problems which arise, especially, late in a contract.

5.4.5 Post delivery support [Topic references C22, C23]

This should be agreed and properly defined in the contract. Support can be expensive, and no Purchaser can realistically expect unlimited support without charge. There should be a contractual limit placed on support, and proper procedures defined for the nature of that support. For example, responsibilities for change control, and problem reporting and correction procedures should have been laid down in the contract for the life of the design/development phase; these should also apply to the support phase. There should also be a mutually agreed procedure for logging of time to be charged to post delivery support, so that disputes may not arise.

- Post delivery support should be treated on the same basis as design/development procedures, and be defined in the contract. It should not be unlimited.

5.5 Liabilities

5.5.1 General liability [Topic reference C15]

Products are expected to perform in a proper manner in the marketplace. This imposes a general liability for consumer protection on the final supplier of the product to the end user. This liability extends to software: there is an inherent requirement in consumer protection legislation for "duty of care" to be exercised by a supplier or manufacturer.

If less than good practice is followed, even where the perpetrator of this practice is convinced that his methods produce a good product, the producer of the product could be exposed to a claim for negligence in the event of a consumer claim for liability.

If software is embedded in a product, as it is in vehicles, it is the constructor of the vehicle who is most likely to be ultimately liable; it is unwise to assume that liability can be readily

passed on to the software producer. Having stated that, however, there is a requirement on the software producer to meet his contractual obligations; if he does not, he may render himself liable for action by the Purchaser.

Legal action of this type is in nobody's best interest: loss of the manufacturer's reputation would still be likely. There is no legal redress for the latter. It is wisest for the Purchaser to ensure that the contract makes such a situation unlikely.

Software—especially commercial packages—is usually **not** sold by its proprietor, but a licence to use it is sold, subject to a range of conditions.

These conditions normally seek to avoid any liability on the part of the software producer for defects in its performance. These clauses do not absolve the software producer from his obligations under Consumer Protection Law so far as a retail purchaser is concerned; they do however prevent an industrial user from bringing a successful action if they do not work as intended—even if there is clear evidence of negligence on the part of the software producer.

Thus, if a subcontractor can successfully show that a problem in his software product was caused by the use of proprietary software there may be no redress for the company placing the contract. This means that it can be very important for a company placing a contract for software design to ensure that commercial software-based tools are specified which are known to be sufficiently free of defects that, if the contractor uses them, they will be unlikely to result in a defective product from the subcontractor.

- Liabilities should be defined as far as possible in the contract, especially where proprietary software is concerned. However, the ultimate liability must be considered to lie with the Purchaser for ensuring that the product is fit for the marketplace, and for ensuring that contracts require adherence to proper practices.

5.5.2 Infringement of patent [Topic reference C4]

This subject is well covered in the references, and these should be consulted. If any doubt exists, legal opinion should be sought.

5.5.3 Consumer protection

Software cannot be **proved** to be 100% free from defect, no matter how much care is taken. There is inevitably some risk of a proprietor of a software-based product attracting a liability suit.

Consumer law requires that a duty of care be exercised; a reasonable defence to a suit may be that "best practice" was followed. (It must be pointed out that that may not be the case outside the UK, where this would only be a defence against negligence. This is the principle of strict liability).

It is suggested that the Purchaser ensures that both he and his contractor follow the procedures recommended by MISRA, including compliance of both organizations with a recognized quality standard for software such as TickIT/ISO 9000.

- It is up to the Purchaser to ensure that both he and the contractor take all the steps recommended by MISRA, and compliance with software standards such as ISO 9000, as appropriate, if the risk of falling foul of consumer protection legislation is to be minimized.

5.5.4 Negligence and insurance [Topic references C17, C18]

The references quoted should be consulted for appropriate advice. It should be remembered that a liability claim can be insured against; subsequent loss of business due to damaged reputation cannot.

5.5.5 Breach of confidentiality [Topic reference C9]

There should be defined procedures in the contract for the exchange of information between contractor and Purchaser; the contractor should be required to sign a confidentiality agreement. If high levels of confidentiality are required, there may be a requirement to restrict numbers of copies, who may hold them, who may authorize what, routes of communication, etc.

However, despite adequate care, confidentiality may still be breached, by accident, or design. If it happens, the contractor, if responsible, may be sued, or the contract terminated. The impact of this on the project must however be fully understood: if for example, the contractor is creating software which is designed to meet a legal deadline for a product—OBD II, say—delay to the project may be out of the question. In practice, suing the contractor may simply create a greater problem for the Purchaser.

The same may also be true where a Purchaser allows confidential details about a contractor to be propagated to a third party. The implication of this may be worse for the Purchaser, though the contractor of course may not wish to lose the Purchaser's business. The best defence against breaches of confidentiality is close control of information, and ensuring proper procedures are adhered to for passage of information. Legal redress is probably inappropriate except in the most serious, probably malicious, cases.

- The best defence against loss of confidential information is close control of the passage of information and authorization. Often, it probably does not lie in legal action.

5.5.6 Disclosure/non-disclosure agreements [Topic reference C21]

Almost any project will involve passing some information to third parties. The contract should make clear procedures for control of information, and authorizations. See above.

5.5.7 Rights of use and access, including service and repair [Topic references C11, C9]

This is of particular relevance in two areas: roadside repair organizations; and franchised/non-franchised dealers.

Vehicle manufacturers often try to restrict servicing of certain items to franchised dealers only. This has the effect that, in the case of software-based systems, their repair by non-franchised organizations becomes near to impossible. This means that roadside repair organizations faced with a defunct vehicle cannot differentiate between fixable mechanical faults, and non-fixable electronic system faults. However, there are companies involved in "chipping" (See Feedback: Support), who appear able to disassemble software in engine management systems, and modify it to increase performance—with impunity.

Further, there are now companies who repair engine management computers, including "fitment" of latest version software; these too, clearly must have some—maybe unauthorized—access to the software in the system. It is believed that there is little that can be done to prevent this type of activity completely, and it is also believed that the law does not offer protection to the owner of the software.

It may be, therefore, more sensible for manufacturers to license "approved" organizations to have access to diagnostic data and maybe even the software itself; this would have to be accompanied by some protection for the manufacturers by controlling access and confidentiality agreements.

- Manufacturers should give consideration to permitting rights of access to at least diagnostic data to "approved" organizations, backed up by confidentiality agreements.

5.6 Deliverables

5.6.1 Project timing requirements phases and milestones [Topic reference C13]

These should tie up with the use of a lifecycle plan (see above). There should be a contractual requirement for approval at each milestone to authorize passage to the next stage of the project.

Before the next stage is entered, there should have been a formal procedure to ensure that the output of the preceding stage meets the requirements, technical and contractual, set by the Purchaser. The Purchaser should appoint an engineer to review, with the contractor, all of the relevant aspects of the project at the point in each stage where approval is to be given for start on the next stage. This type of procedure allows for corrective actions to be initiated and carried out before the project goes seriously wrong in the event of a problem occurring.

There is often pressure on engineers to meet impossible targets for software development

projects; this can all too easily be transferred to the contractor. The latter will hardly turn down business because the project timing requirements are unrealistic—but this acceptance may indeed result in serious problems occurring later in the project. Part of the reason for pressure on timing requirements is simply that in the vehicle industry software is often poorly understood by programme managers, who have often had little exposure to the discipline of software engineering. The apparent cost of a formalized procedure for developing software is perceived as too high; yet, when the project goes wrong, the cost of fixing it, especially the software, is largely accepted.

At least, adherence to a lifecycle plan with a requirement for approval to proceed at each stage will show very quickly how the money spent is allocated, and aid in a more realistic view of software development to be taken in the future.

For this type of approach to be effective, however, the deliverables from each stage must be clearly defined in the contract, and agreed by both parties.

- Use of a lifecycle plan aids in understanding where money is spent in software projects, especially if each stage must result in approval being required for start of the next stage. This is only effective if the deliverables from each stage are clearly defined in the contract.

5.6.2 Transfer of design data—media requirements [Topic reference C25]

The contract should define how design data is to be transferred to the contractor, and how the finished design is to be transferred to the Purchaser. The deliverables will usually be paper—software source listings, flow diagrams, architecture diagrams; version and change control records; problem reporting and actions records; personnel timing records; plus delivery in other media. The latter will often be, in the case of software, floppy discs or magnetic tapes, but may include diagrams and text sent via electronic mail, and EPROMs containing data or program. The formats of all data exchanges must be defined in the contract, including, where documentation is concerned, word processor types, and graphic or CAD formats. The number of copies should be stipulated, and to whom they are to be delivered. There may be a stipulation that all magnetic or electronic copies and even paper copies of data and documentation relating to the project are destroyed or handed over by the contractor. All of such requirements **must** be contractually defined.

- All requirements for transfer of data and documentation, magnetic, electronic and paper must be defined in the contract, along with the number of copies, personnel receiving copies, and any requirements for retention or destruction of residual data.

5.6.3 Software cost estimating [Topic reference C26]

This is notoriously difficult.

All software contracts should contain a contingency arrangement to cover for unforeseeable

difficulties. If the contractor includes to great a contingency in his tender, it may be uncompetitive; too little, and he risks losing money if a problem occurs.

A staged contract is usually advantageous, with approvals at each stage: if difficulties become obvious they are likely to be addressed before they become a serious problem, rather than the project proceeding to the stage where the contractor runs out of money, and either asks for more, or else, having perceived the problem, takes liberties which affect the quality of the product.

- Software cost estimates are a well recognized area of difficulty. Staged contracts are a significant help in obviating problems, by identifying them at an early stage, before the project has gone seriously wrong.

5.7 Miscellaneous contract issues

5.7.1 Standard contracts [Topic reference C1]

Standard contracts are usually not well aligned with the contracting of software. They will almost always require Annexes to define special conditions and technical contract issues. It is most important to recognize the issues which are special to software, and take account of them in any software contract. References are given which offer model contracts as well as advice for producers of contracts in the software arena; it is strongly suggested that these references are consulted.

- Standard contracts should be treated with caution where software is involved, and advice should be sought from the references. There are also model contracts referred to which are suitable for software projects. Advice should sought from an expert in software contracts. The advice of software engineers is also important.

5.7.2 Contracts placed outside national boundaries

It should be borne in mind that other countries' laws relating to contracts may be quite different to UK law, particularly in respect of liabilities. Advice should be sought from legally qualified persons on this aspect, with special expertise in software contracts.

- Seek advice from legally qualified persons with special expertise in software contracts regarding international contracts. Advice of software engineers is also important.

5.7.3 Third party support and modification [Topic reference C24]

If it is desired to involve third parties for support of software, including modifications, in the field, this should be provided for in the contract. The responsibilities and rights of the third parties, as well as those of the contractor and Purchaser should be spelled out. It should also

be borne in mind that third party support may not be desirable unless the third party(ies) has been involved and fully informed regarding technical aspects of the software (and indeed, hardware) during the development phases. This may well be a contentious matter for the contractor—and sometimes, even for the Purchaser's own staff—so it strongly advised that there is full agreement on the mechanism to be adopted before the contract is created.

This issue can also arise for other reasons: for example, if the contractor is deemed to have failed in his duties, or has ceased trading at some point during the life of the product. For protection of the Purchaser, in either event, the use of escrow may be demanded (see earlier section on this), but it should be recognized that it is not an easy matter for a third party to just carry on where the original one left off.

Another third party issue is third party verification and validation. Clearly, if a third party is to be employed to oversee the output of various stages of a project, or to carry out independent verification and validation procedures, then that third party must have both total access to the project technical data, and procedures, if the audit is to be effective. This raises a serious issue of confidentiality, which may be very contentious to a contractor. This issue must be agreed at the earliest stage with all the parties involved, and properly defined in the contract. The least contentious way of involving an independent audit is probably by audit of quality systems of the contractor(s), and this may be stipulated as a condition of tender rather than in an individual contract; nevertheless, there may be a need for a conformity inspection at some point, particularly if a technical problem arises.

- Agree issues relating to third party audit, support and modification at the earliest stage possible, including confidentiality matters, and ensure that any such requirements are spelled out clearly and unambiguously in the contract.

5.7.4 Retention of records post delivery, access, and security [Topic reference C25]

The Purchaser may not wish a contractor to hold any records of his work after completion of a contract, or alternatively may require that they are held for a specified time period after acceptance into service of the product. Clearly, there is some risk in a contractor alone being required to retain records; in the event that the contractor ceases trading, the Purchaser may not easily gain access to the records. In addition, however records are retained, there may be a risk that they will be communicated, mistakenly, or maliciously, to a third party.

The terms of any retention of records, and security arrangements, should be defined in the contract, together with a mechanism for access to those records, authorization procedure, routes of communication, notice for requiring access, etc. Normally, it is wise to retain records for at least ten years after cessation of production; some companies have found it useful to retain records for longer still.

- Ensure that requirements for retention of records are fully laid down in the contract, including stipulated time period and media.

5.7.5 Precedent

Much UK law is heavily influenced by precedent. There appears, however, to be little, if any, precedent regarding consumer protection law, and especially, software aspects. Some issues regarding software are well covered by existing law and precedent, e.g. copyright law, but other issues are less clear. It is imperative to take legal advice if unsure, and also important to recognize that, if one becomes involved in legal action in respect of software, that one could be making legal history.

- Legal issues regarding software appear to be less subject to precedent than many other legal matters. If unsure, take legal advice, do not rely on experience of non-software contracts only.

6. Conclusions and comments

- Ensure that relationships of all parties to a software contract agree its provisions, and define all requirements, obligations, and rights of all parties in the contract.

- Ensure that adequate safeguards and contingencies are built into the contract to cover the possibility of technical problems.

- Punitive provisions in a contract such as penalty clauses may be counterproductive, and in the extreme may cause the contractor's business to fail: the emphasis should be on recognizing and correcting problems as early in the lifecycle as possible.

- Ensure that all deliverables are properly defined.

- Do not rely on precedent as it exists for other types of contract—it may not apply to some aspects of software.

- If in **any** doubt, **seek legal advice** from a source with experience of software contracts!

Appendix A. Subcontract reference documents

	Title	Abbreviation
1.	IEE/IMechE *General Conditions of Contract*	MF/1, MF/2
2.	IEE *Product Liability* Brief	IEE
3.	Computer Services Association SCS Notes	CSA
4.	*FIDIC Conditions*, J. Sawyers & C. Gillott	FIDIC
5.	NCC *STARTS Guide* (last published 1989)	STARTS
6.	*Software Engineering*, I. Summerville	SE
7.	*Intellectual Property*, D. Bainbridge	IP
8.	*Software and Copyright Law*, D. Bainbridge	S&CL
9.	*Computer Contracts*, R. Morgan & G. Stedman	CC
10.	European Space Agency (ESA) PSS-05	ESA
11.	IEC WG9/WG10	IEC
12.	*Diagnostics and Integrated Vehicle Systems*, MISRA Report 1, 1994.	
13.	*Integrity*, MISRA Report 2, 1994.	
14.	*Software in Control Systems*, MISRA Report 4, 1994.	

Appendix B. Software subcontracting in the UK civil nuclear and aerospace industries

B.1 Introduction

The purpose of this Appendix is to describe the process used for both customers and vendor when subcontracting tasks for software. The industries of interest within this report are those of the UK civil nuclear and UK aerospace industries.

For each of the industrial sectors described above, the contractual and technical aspects will be examined. In addition, the key product characteristics will be identified for each sector.

B.2 UK civil nuclear

Within the civil nuclear sector, software systems fall into one of three broad categories:

(a) Control Systems

(b) Protection Systems

(c) Monitoring/Information System

Due to the need to satisfy the Nuclear Installations Inspectorate (NII) that the plant is safe, all system design activities feed into a safety case to some extent. The degree to which a particular subsystem is relied on by the safety case will depend on the safety argument. A safety critical subsystem will require detailed reference in the safety case and the design will be closely scrutinized by the assessor. For this reason, the required quality assurance regime placed on the vendor can be both stringent and detailed. Where not only the deliverables are defined by the customer, but also the design process itself. Second party quality audits are often conducted to ensure that the vendor is complying with the contract requirements. The customer is also likely to review and approve key design documentation. As a result of these activities subcontracting is structured such that the customer has a clear view of both quality assurance and contact progress. The software is developed to the IEC 880 standard.

When requesting "Invitations to Tender" from possible vendors, a part of the assessment criteria is often the calibre of staff and the skills/tools possessed by the organization. Sample CVs of representative staff is often requested, and if a vendor is successful, named project personnel are often written into the contract.

The use of penalty clauses are used extensively within contracts, however there is also an awareness that managing the subcontractor to achieve a quality product is a better approach as it reduces the risk of failure. Placing heavy penalties on a vendor when he cannot tolerate them may accelerate failure of the project, once a problem is encountered.

The high capital cost of many software based systems and the degree of rigour within the quality control process, often results in the customer accepting contractual milestones as the delivery point, where part payment can be awarded. This provides the vendor with means of managing project financing and provides the customer with the opportunity to assess and witness product design and testing.

B.3 Aerospace

The aerospace market is very broad and the quality requirements vary from a good commercial standard to mission critical. Subcontracts are placed in many ways, dependent on the business and the customer/vendor relationship. Here, the "hands-off" approach and customer involvement approach are both used.

Factors which affect the subcontracting arrangements include:

(a) Whether the product is bespoke or based on existing components.

(b) Whether the contract is the first with the vendor or a follow-on contract.

(c) Whether certification is required by FAA/CAA.

Some systems are purchased as a "black box" in a similar manner to the automotive industry. However, within the civil aerospace engine sector, memoranda of understanding have been used in the past to create close relationships between customer and subcontractor. This was often done due to the requirement of the customer for guaranteed close subcontractor support over a long period of time. This work is often performed on a "cost plus profit" basis. This is now changing, as customers are seeking to share more of the risk with their subcontractors. This is particularly the case during the development of new products. A close relationship is still required during the life of the product as both customer and contractor produce variants of the core product through life. In some instances, the customer will retain the task of programming the *raison d'être* functions, such as the "engine laws" for a gas turbine product. This holds many similarities with the automotive industry, particularly with engine management systems.

Appendix C. Subcontracting in the process industry

C.1 Introduction

This Appendix discusses the subcontracting of software within the process industry based on experience. As a result of the scale of the industry and the various sizes of the companies in the industry the report is of a general nature and no references to companies or projects are made.

C.2 Subcontracting issues

From experience, in most circumstances a customer will be seeking to purchase a complete system, which may contain software, rather than actual software. One exception to this general case is where the customer is seeking to update or extend the capability or functionality of existing equipment. This revision is therefore different in that it is seen as an extension rather than a new system.

Indeed, many of the systems being procured by the industry are based around Programmable Logic Controllers (PLCs), which have their own particular software programming languages. The more traditional style of software development, using the common high and low level languages, tend to be implemented by the manufacturers of the PLCs, leaving customers and suppliers of PLC equipment to converse in the PLC language, for example ladder logic.

C.3 Tendering

In procuring a system many customers issue an Invitation To Tender (ITT) to those suppliers believed to be capable of producing the equipment.

The requirements of the ITT are usually presented in a User Requirements Specification, or similar document. This type of requirement specification, written in English, defines the functional and performance attributes of the desired system.

The detail to which the requirements are presented tend to be driven by the availability and complexity of the system. If the company is simply seeking a basic system but with specific performance criteria the requirements will be less than if the system is somewhat unique.

If there are any safety requirements of the system then these will be clearly specified. When the contract is let the customer may specify safety related functions in the form of Cause and Effect Charts. The Cause and Effect Chart, is a form of specification specific to the process industry. The chart takes the form of a matrix which cross references input signals to desired actions, for example what happens if a signal from a fire detector is raised.

	Action				
Input Signal					
	n		n		n
		n	n		

Generally the customer will not specify "how" the system should be implemented, only "what" his requirement. Instead the customer will leave it to the supplier, based on his familiarity with certain equipment, to provide a solution. Only if the customer has very precise requirements, for example the system must be based on Triple Modular Redundancy equipment, will he direct the supplier towards a definitive solution.

In response to the ITT the supplier will define how his system, which may be based on common PLCs, will meet the stated functions. The customer then reviews the ITT responses and select the most appropriate tender. Such reviews may take the form of a defined evaluation marking scheme.

There has been increased attention given to supplier's software capability especially where a more traditional software development is being proposed for example non-PLC. This is as a result of companies experiencing software oriented problems during commissioning. Some companies which have been responsible for designing good mechanical/electrical systems have underestimated the impact of software on their products.

In some circumstances the customer may go direct to a particular supplier or manufacturer because they are the only supplier or manufacturer capable of providing the required system. Indeed some customers have a single supplier agreement based on commercial considerations. Companies spend a vast amount of money in training their staff to operate and maintain the plant. Clearly such costs can be minimized if common equipment is purchased instead of new and unfamiliar systems.

Over recent years there has been a tendency by the industry, in line with other industries, to try to get the design requirements correct early in the lifecycle. Thereby reducing the level of changes late in the development process when the cost of any change is greater. Indeed, the payment terms defined in the contract may be aligned with lifecycle phases.

C.4 Consultants and software houses

One exception to the subcontracting of systems rather than software is where the customer is developing his own systems and may, as a result of lack of experience in house, employ a software house or contractor to produce the software. However, the stages following are similar to the steps in subcontracting systems.

C.5 Commercial issues

The commercial aspects are dealt with in one of two ways.

Companies may issue their own standard terms and conditions. Any of the terms which the supplier does not approve are then discussed.

Alternatively, as part of the ITT response the supplier maybe required to submit his own terms and conditions. These terms and conditions will be reviewed by the customer during the contracting phase. Again any terms which are not acceptable will be discussed.

Specific commercial requirements, in both cases, will be defined in the actual contract.

The companies do not appear to use any general terms and conditions as promoted by the industry or Institutions.

C.6 Conclusions

Companies within the process industry tend to contract systems rather than software, although software considerations are appearing in the procurement exercise, for example tender documentation.

The way in which the system requirements are defined is similar to other industries except that the "how" is more commonly left to the supplier. Only in special circumstances will precise methods of implementation be specified.

Systems tend to be based on equipment such as PLCs which only require programming at a high level language.

The tendering and contracting exercise is again similar to other industries, and is centred around some form of lifecycle. Since projects tend to be long term, the defined lifecycle forms the foundation of a payment schedule. The supplier is paid on achieving certain events, for example Factory Acceptance Test where the supplier demonstrates the system meets its requirements prior to its release.

The terms and conditions applied to contracts may be standard to the customer, or to the supplier. Any terms which are special to the project are specified in the contract. Such defined terms will override standard terms and conditions.

Appendix D. Applicability of standards and model contracts

The following marking scheme applies in the Tables that follow, with the mark given in square parentheses […].

- limited
- useful
- relevant.

TOPIC	MF/1	ISO 9000-3	SCS Guidance Notes	IEC 65A(Sec) 122	STARTS Guide	IEE Legislation Brief
A.1 Customer requirements specification	CL 13 [Useful]	5.3 [Limited]	GN02 [Limited]	CL 9 CL 10 CL 12 [Relevant]	CH 10 App B [Relevant]	Sections 5.13 - 5.20 [Limited]
A.2 Adoption of an agreed lifecycle plan	CL 14 [Useful]	5.1 [Limited]		CL 8 [Relevant]	CH 3 CH 7-2.4 Fig 13.1 [Relevant]	
A.3 Use of proprietary standard software				CL 11 [Relevant]	CH 9-5.5.2	
A.4 Reuse of the suppliers' existing software				CL 11 [Relevant]	CH 9-5.5.2	
A.5 Division of integration responsibilities	CL 2.3 & 3 [Relevant]		GN05 [Relevant]	CL 11 CL 13 [Relevant]	CH 4-6.6 [Limited]	
A.6 Acceptance responsibilities	CL 2 [Limited]	4.1.2 5.2.2d) 5.8 [Useful]			CH 5-5.4.5 CH 7-5.9 CH 12-6 CH 12-2 App B-9 [Relevant]	
A.7 Progress reporting	CL 14.6 [Limited]	5.10.5 [Limited]	GN08 GN09 [Relevant]	CL 11 [Relevant]	CH 4-5.3 [Limited]	
A.8 Maintenance	CL 15.6 [Limited]	5.10 [Relevant]	GN11 [Useful]	CL 11 CL 17 [Relevant]	CH 13 CH 5-5.3.4 CH 6-3.4 CH 9-3.14 CH 11-3.4.6 [Relevant]	

TOPIC	MF/1	ISO 9000-3	SCS Guidance Notes	IEC 65A(Sec) 122	STARTS Guide	IEE Legislation Brief
B.1 Standards		[Relevant]	GN02 [Limited]	CL 11 CL 18 [Relevant]	CH 9-3.15 CH 5-3.8 CH 5-4.1 [Limited]	
B.2 Quality Assurance and accreditation		[Relevant]	GN05 [Useful]	CL 8 CL 12 CL 14 CL 16 [Relevant]	CH 5 CH 7-4.5 CH 9-3.12 [Relevant]	
B.3 Qualifications, competence and training		6.9 [Limited]	GN02 GN03 GN06 GN07 [Relevant]	CL 7 [Relevant]	CH 5-4.10 CH 12-12.2 CH 7-4.3 [Limited]	Sections 1 and 5.6 [Limited]
B.4 Management and key personnel responsibilities and accountability	CL 2 [Limited]	4.1 [Relevant]	GN05 GN07 GN12 [Relevant]	CL 7 [Useful]	CH 4 [Useful]	Section 1 [Limited]
B.5 Requirements for special tools		5.2.2e) [Limited]		CL 11 [Limited]	CH 6-1 CH 12-4.8 CH 4-5.8 [Useful]	
B.6 Programming language requirements		5.6.3a) [Limited]		CL 11 [Relevant]	CH 7-6.4	
B.7 Compiler and tool validation				CL 11 CL 14 [Relevant]	CH 7-6.4	

TOPIC	MF/1	ISO 9000-3	SCS Guidance Notes	IEC 65A(Sec) 122	STARTS Guide	IEE Legislation Brief
B.8 Process and product metrics for reports		6.4 [Relevant]			CH 8 [Useful]	
B.9 Problem reporting procedures		4.4 [Useful]	GN09 [Useful]		CH 4-5.5 CH 7-4.6 CH 6-2 CH 13-2 [Useful]	
B.10 Joint design reviews		4.1.3 5.4.6a) [Useful]			CH 5-4.9 [Limited]	
B.11 Independent assessment/audit			GN12 [Relevant]	CL 11 CL 12 CL 14 CL 15 [Relevant]	CH 7-4.4 [Limited]	

TOPIC	MF/1	ISO 9000-3	SCS Guidance Notes	IEC 65A(Sec) 122	STARTS Guide	IEE Legislation Brief
C.1 Use of standard conditions of contract	[Relevant]		GN03 [Relevant]		CH 11-3.4 [Useful]	
C.2 Intellectual property rights	CL 42 [Useful]				CH 11-3.4.2 [Limited]	
C.3 Copyright	CL 42 [Relevant]				CH 11-3.4.2 [Limited]	
C.4 Patents	CL 42 [Relevant]				CH 11-3.4.3 [Limited]	
C.5 Knowhow			GN06 GN07 [Limited]			
C.6 Ownership of supplied and delivered materials			GN05 [Limited]		CH 11-3.4.2 [Limited]	
C.7 Licences for proprietary software and protocols					CH 9-7	
C.8 Transferability of licences						
C.9 Right of use, access and confidentiality agreements	CL 15.4 [Limited]		GN05 [Limited]		CH 11-3.4.8	
C.10 Third party retention of software material - escrow					CH 12-11.8 [Useful]	

TOPIC	MF/1	ISO 9000-3	SCS Guidance Notes	IEC 65A(Sec) 122	STARTS Guide	IEE Legislation Brief
C.11 Access to data for diagnostics and service				CL 14 [Useful]		
C.12 Disclosure/non-disclosure agreements on third party/ supplier chain involvement			GN05 [Limited]			
C.13 Project timing requirements, including phases and milestones				CL 11 [Useful]	CH 4-6 CH 11-3.5 [Limited]	
C.14 Deliverables, including phase deliverables	CL 24 [Limited]				App B-7 CH 5-5.4.6 CH 6-2.5	
C.15 Liability	CL 36,41, 44 [Relevant]		GN01 GN02 GN04 [Relevant]		CH 9-5.2 CH 9-6.1	Sections 3, 4,7,8 & 9 [Relevant]
C.16 Customer protection			GN02 GN03 [Relevant]		App C-C.2.2 [Limited]	Section 6 [Relevant]
C.17 Negligence			GN01 GN03 GN04 [Relevant]			Sections 7 & 9 [Relevant]
C.18 Insurance	CL 47 [Relevant]		GN04 [Relevant]			Section 10 [Relevant]
C.19 Warranty commitments					CH 12-10.8 CH 11-3.4.5 [Limited]	Section 10 [Relevant]

TOPIC	MF/1	ISO 9000-3	SCS Guidance Notes	IEE 65A(Sec) 122	STARTS Guide	IEE Legislation Brief
C.20 Performance bonds	CL 8 [Relevant]					
C.21 Penalty clauses					CH 9-3.9	
C.22 Post-delivery support			GN11 [Relevant]		CH 9-3.14 App B-6.4	
C.23 Responsibilities for post-delivery modifications		5.5.2e) [Limited]	GN11 [Relevant]			
C.24 Third party support and modification			GN03 GN011 [Useful]			
C.25 Retention, security and access to records post-delivery			GN10 [Relevant]	CL 7.2.6 [Useful]		
C.26 Software cost estimating methods					CH 6-2.2 App E [Limited]	